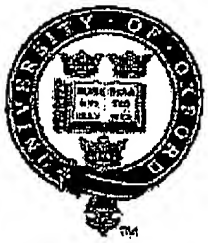

OXFORD UNIVERSITY COMPUTING LABORATORY

courses | research | people | about us | news

Overview: Hardware Compilation and the Handel-C language

Christian Peter, Oxford University Computing Laboratory, UK
cpeter@comlab.ox.ac.uk

Hardware compilation [Page 1996] is a powerful tool for hardware/software codesign. Conventionally, it is common to draft a system using a high-level programming language, like C, and then realise the hardware part of that system using a hardware description language like VHDL [Perry 1994]. Hardware compilation using Handel-C [Page 1996] simplifies the design process, since it provides the convenience of a C-like high-level programming language to generate the hardware as well.

Handel-C is a programming language designed for compiling programs into hardware images of FPGAs or ASICs. It is basically a small subset of C, extended with a few constructs for configuring the hardware device and to support generation of efficient hardware. It comprises all common expressions necessary to describe complex algorithms, but lacks processor-oriented features like pointers and floating point arithmetic. The programs are mapped into hardware at the netlist level, currently in xnf or edif format.

Opposed to other approaches of high-level language hardware design, which actually use C to describe the behaviour and simply translate it into a netlist, Handel-C targets hardware directly, and provides a few hardware optimising features. A big advantage, compared to the C-translators, is that variables and constants can be given a certain width, as small as one bit. When using C as the describing language, the smallest integer is possibly 8 bits wide, if not 16, which means that one wastes at least 7 registers when declaring a simple flag. Also, Handel-C provides bit manipulation operators and the possibility of parallel processing of single statements or whole modules. This can not be realised with other approaches basing on a sequential language.

Comparing Handel-C with VHDL shows that the aims of these languages are quite different. VHDL is designed for hardware engineers who want to create sophisticated circuits. It provides all constructs necessary to craft complex, tailor made hardware designs. By choosing the right elements and language constructs in the right order, the specialist can specify every single gate or flip-flop built and manipulate the propagation delays of signals throughout the system. On the other hand, VHDL expects that the developer knows about low-level hardware and requires him continuously thinking about the gate-level effects of every single code sequence. This quite easily distracts from the actual algorithmic or functional subject and ties up a lot of the designer's attention.
In contrast to that, Handel-C is not designed to be a hardware description language, but a high-level programming language with hardware output. It doesn't provide highly specialised hardware features and allows only the design of digital, synchronous circuits. Instead of trying to cover all potentially possible design particularities, its focus is on fast prototyping and optimising at the algorithmic level. The low-level problems are hidden completely, all the gate-level decisions and optimisation are done by the compiler so that the programmer can focus his mind on the task he wants to implement. As a consequence, hardware design using Handel-C resembles more to programming than to hardware engineering, and in fact, this language is developed for programmers who have no hardware knowledge at all.

That Handel-C is designed for fast prototyping can also be seen when working with the simulator. This small but efficient tool simulates the behaviour of the circuit at the algorithmic level, based on the semantics of the Handel-C program. Compared with hardware simulators, which usually analyse the gate-level function of the circuit, the simulation time is very short (seconds opposed to several minutes). Together with the very fast compilation, this encourages the designer to try out several implementations of the design.

To sum up, hardware design with Handel-C is very much like programming. Unlike with hardware description languages, the designer is not confronted with gate-level problems like fan-in and fan-out or choosing the appropriate type of gates or registers to be used. Apart from freeing the programmer's mind from those low-level decisions, it is much faster and more convenient to describe the system's desired behaviour at the algorithmic level. The fast compilation, combined with the high-level simulator, allows to try out several implementation strategies within a very short time.

[Page 1996] Page, I. : Constructing hardware-software systems from a single description ;
Journal of VLSI Signal Processing, 12(1), pp. 87-107, 1996.

[Perry 1994] Perry, Douglas L . : VHDL, 2nd edition ;
McGraw-Hill series on computer engineering) ; New York, 1994

back to text

Home